Numerical Treatment of PDE

- here: restriction to the elliptic case, Poisson equation
- several principal approaches:
  - **finite differences (FD)**: direct approximation of each derivative (cf. ODE)
  - **finite volumes (FV)**: direct implementation of conservation laws on small volumes; esp. for flows
  - **finite elements (FEM)**: variational approach, study some weak form of the PDE
- rough characterization:
  - FD: straightforward, easy to implement, poor theoretical background
  - FEM: more complicated to implement, but rich mathematical theory available

Finite Difference Methods 1

- Place a (in general regular) grid on the given domain.
- Replace derivatives by difference quotients:
  - first derivatives: **forward**, **backward**, or **central** differences

$$\frac{\partial u}{\partial x}(\xi) \doteq \frac{u(\xi+h_x)-u(\xi)}{h_x}, \quad \frac{u(\xi)-u(\xi-h_x)}{h_x}, \quad \frac{u(\xi+h_x)-u(\xi-h_x)}{2h_x}$$

  - second derivatives: standard **3-point-stencil**

$$\frac{\partial^2 u}{\partial x^2} \doteq \frac{u(\xi+h_x)-2u(\xi)+u(\xi-h_x)}{h_x^2}$$

  - Laplacian in 2D or 3D, resp: 5-point or 7-point stencil, resp.
- in each inner grid point: set up a *difference equation*
  - in each inner grid point: one *degree of freedom* per (scalar) unknown function
  - concrete equation near boundary depends on bound. cond.

Finite Difference Methods 2

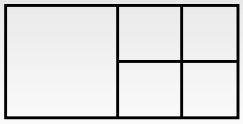
- a simple example: Poisson equation on unit square

$$-\Delta u = f \quad \text{on} \quad]0,1[^2$$

- use an equidistant square grid with $h = h_x = h_y = 1/N$
- number of d.o.f./unknowns: $M = (N-1)^2$
- linear system Ax=b with
  - MxM-matrix A (the difference equations)
  - M-vectors b (right-hand side) and x (desired values of u)
- A is a *sparse matrix* (band structure):
  - **Dirichlet b.c.**: all diagonal elements are 4, per row between 2 and 4 non-zeroes (-1); boundary values to right-hand side
  - **Neumann b.c.**; diagonal elements are 2 or 3 near the boundary and 4 elsewhere; a corresponding number of –1's (2 to 4) in each row; pairs (1,-1) along boundary to r.h.s.

Finite Difference Methods 3

- order of accuracy: $\left\| u^{\text{computed}} - u^{\text{exact}} \right\| = O(h^2) = O(N^{-2})$
- *curse of dimension:* for that, we need $O(N^d)$ points
- possibilities of an improvement:
  - use higher-order stencils (involving more than two neighbouring grid points in each direction): matrix gets a denser structure
  - use a finer or a locally refined (*adaptive*) grid (problem: what to do with *hanging nodes* where coarser and finer subregions meet?)



Finite Element Methods

- no direct discretization of derivatives, but use some *weak form* of the PDE instead of the PDE itself
- five basic steps:
  - **subdivision** or **grid generation**: decompose the underlying domain into single elements of finite size
  - **weak form**: the PDE has no longer to be fulfilled in each point of the domain, but only w.r.t. some scalar product
  - **finite dimensional ansatz space**: in the weak form, replace the continuous solution by suitable finite-dimensional approximations
  - **system of linear equations**: construct the corresponding system of linear equations (one equation for each degree of freedom)
  - **solution of the linear system**: use appropriate solvers in order to obtain the finite element approximation to the PDE's solution

Subdivision, Grid Generation

- subdivide the problem's domain into *finite elements*:
  - from statics/mechanics/civil eng.: decompose complex objects into standard parts whose behaviour is easier to describe, and derive the object's behaviour from that
  - in 3D, we get a finite element mesh consisting of
    - **elements**: 3D atoms (cubes, tetrahedra, ...)
    - **faces**: 2D surface structures of elements (triangles, squares,...)
    - **edges**: 1D boundary structures of elements
    - grid points or **nodes**: where the unknowns are defined
- associated to each grid point is an *ansatz function* φ_k
  - finite support: nonzero only in neighbouring elements
  - all ansatz functions together span the linear and finite-dimensional *ansatz space,* of which they form a basis
  - in this ansatz space V_n, look for an approximation to the PDE's solution

Weak Form of the PDE

- let L denote the differential operator (e.g. Laplacian)
- instead of $Lu = f \text{ on } \Omega$, consider

$$\int_\Omega Lu \cdot \psi_l \, d\Omega = \int_\Omega f \cdot \psi_l \, d\Omega \quad \forall \psi_l$$

for some set of *test functions* ψ_l

- method of **weighted residuals** or **Galerkin approach**
- a second finite-dimensional linear space spanned by these test functions – the *test space* W_n
- identical test and ansatz spaces: **Ritz-Galerkin approach**
- different test and ansatz spaces: **Petrov-Galerkin approach**
- due to linearity: require weak form for basis functions only
- introducing a *bilinear form a(.,.)* and a *linear form b(.)*:

$$a(u, \psi_l) = b(\psi_l) \quad \forall \psi_l \in W_n$$

Discrete Approximation and Equations

- In the above equation, replace the (exact/continuous) solution u by a discrete approximation in V_n:

$$u_n = \sum_k \alpha_k \varphi_k$$

- in the weak form:

$$a(u_n, \psi_l) = a\left(\sum_k \alpha_k \varphi_k, \psi_l \right)$$

$$= \sum_k \alpha_k \cdot a(\varphi_k, \psi_l) = b(\psi_l) \quad \forall \psi_l \in W_n$$

- neither the a(.,.) nor the b(.) depend on the respective approximation to u, but only on the problem!
- compute them once for all at the beginning: leads, as expected, to a system of linear equations Ax=b with the so-called *stiffness matrix A*

Solving the Resulting SLE

- properties of A and of our system of linear equations:
  - esp. in the Ritz-Galerkin case $V_n = W_n$, A is often SPD
  - dream situation: A is diagonal, i.e. all ansatz/test functions are (bi-) orthogonal (rare or hard to achieve, resp.!)
  - due to the ansatz/test functions' finite support, A is typically *sparse* (due to this and due to A's size in practical computations, we need iterative solvers):

$$a_{i,j} = a(\varphi_i, \varphi_j) = \int_\Omega L\varphi_i \cdot \varphi_j \, \mathrm{d}\Omega = 0$$

    for non-overlapping supports!
- strategy hence:
  - choose ansatz/test spaces with good approximation properties
  - construct bases of these which result in „nice" matrices and, thus, in „fast-to-solve" SLE (cf. *hierarchical bases*)